Working Papers

Technische Hochschule Ingolstadt

Zukunft in Bewegung

Technische Hochschule Ingolstadt

Working Paper

Heft Nr. 56 aus der Reihe
„**Arbeitsberichte – Working Papers**"

ISSN 1612-6483
Ingolstadt, im Dezember 2020

Prof. Dr.-Ing. Wolfgang Krämer

Modellprädiktive Regelung einer Drohne

Abstract

Bericht über die Ergebnisse eines Forschungsfreisemesters

Einleitung

In diesem Bericht stelle ich die während meines Forschungsfreisemesters im Sommer 2020 am Department of Chemical Engineering der University of California – Santa Barbara erzielten Ergebnisse vor. Im Rahmen dieser Arbeiten sollte die Anwendung der Methode der prädiktiven Regelung (MPC – Model Predictive Control) auf die Regelung einer Drohne untersucht werden. Insbesondere wurde das Ziel verfolgt, entsprechende Versuche für das regelungstechnische Praktikum zu entwickeln.

Die Regelung einer Drohne ist ein sehr attraktives Anwendungsbeispiel für ein Praktikum, da die Drohne im Vergleich zu den im chemical Engineering üblichen Beispielen, wie Füllstands- und Temperaturregelungen, ein schnelles dynamisches Verhalten zeigt und da man die Auswirkungen der Regelung auf die Bewegungen der Drohne direkt sehen kann. Für das Praktikum steht die Drohne crazyflie 2.1 zur Verfügung [1].

Für die Regelung soll das Programmpaket mpctools verwendet werden, das an der University of Wisconsin – Madison entwickelt wurde [2]. Es stellt einerseits eine Toolbox (Erweiterung) für Matlab dar, sodass die bekannte Bedienoberfläche und Funktionalität von Matlab zur Verfügung stehen, und andererseits ein API (Application Programming Interface) für das Optimierungsprogramm casadi [3], das für die im Rahmen von MPC erforderlichen Optimierungen eingesetzt wird.

Wegen der Einschränkungen infolge der Covid-19-Pandemie waren nur wenige experimentelle Untersuchungen möglich. Der Schwerpunkt der Arbeiten lag daher auf Simulationsuntersuchungen.

Zunächst wurde ein dynamisches Modell der Drohne erstellt und in eine Matlab-Funktion umgesetzt. Sodann wurde die Anwendung der bekannten Methode der quadratisch optimalen Regelung (LQR – Linear Quadratic Regulator) untersucht. Da hierfür die Kenntnis aller Zustandsgrößen erforderlich ist, wurden in einem nächsten Schritt verschiedene Methoden der Zustandsschätzung erprobt. Schließlich wurde eine Trajektorienplanung für die Drohne mit MPC entwickelt; in diesem Zusammenhang wurde auch ein Vorschlag für eine zeitoptimale Trajektorienplanung ausgearbeitet. Außerdem wurden einfache experimentelle Untersuchungen zum Messrauschen der Sensoren der Drohne durchgeführt.

Die Trajektorienplanung erfolgt so, dass mittels mpctools in Matlab eine gewünschte Trajektorie als Folge von Sollwerten berechnet wird; hierbei werden alle Anforderungen an die

Trajektorie, wie z.B. das Umgehen von Hindernissen und die Einhaltung von Stellgrößenbegrenzungen, berücksichtigt. Die Sollwertfolge wird in einer Datei abgelegt und vom sog. Commander, einem Programm zur Übermittlung von Befehlen an die Drohne, eingelesen. Der Commander sendet dann die berechneten Sollwerte in Echtzeit an die Drohne, die über eine Onboard-Regelung verfügt.

Diese Methode konnte erfolgreich im Labor erprobt werden. Damit steht eine gute Möglichkeit zur Verfügung, MPC für die Regelung der Drohne im Praktikum anzuwenden.

Die Ergebnisse der o.g. Schritte wurden jeweils in kurzen Statusberichten oder Präsentationen zusammengefasst. Diese Präsentationen und Statusberichte sind im Folgenden angefügt und stellen den Hauptteil dieses Gesamtberichts dar:

- Präsentation: Model of the Drone crazyflie 2.1
- Status Report 1: State Feedback
- Status Report 2: State Estimation
- Status Report 3: Structure of LQR Feedback Matrix, Constraints, and Madgwick Filter Continued
- Note: First Experimental Results
- Präsentation: Path Planning for the Drone crazyflie 2.1

Danksagung

Zunächst danke ich meinem Kollegen Prof. Dr. James B. Rawlings von der University of California – Santa Barbara. Er hat mir nicht nur dieses interessante Projekt im Rahmen meines Freisemesters ermöglicht, wodurch ich einen guten Einblick in die Methode der prädiktiven Regelung bekommen konnte, sondern mich auch unterstützt, gut durch die Covid-19-Krisenzeit zu kommen.

Des Weiteren danke ich Koty McAllister, Doktorand bei Prof. Rawlings, für seine vielfältige Unterstützung bei Problemen mit mpctools und bei den experimentellen Untersuchungen; diese wären ohne seine sorgfältige Vorbereitung in der zur Verfügung stehenden Zeit nicht möglich gewesen.

Mein besonderer Dank gilt meinen Kollegen an der TH Ingolstadt, die meine Lehrveranstaltungen im SS 2020 übernommen und mir so den Auslandsaufenthalt ermöglicht haben. Außerdem bin ich der Fakultät für Maschinenbau und der Hochschulleitung zu Dank für Ihre Unterstützung und Genehmigung meines Vorhabens verpflichtet.

Literatur

[1] https://www.bitcraze.io/products/crazyflie-2-1/, aufgerufen am 22.04.2020

[2] https://bitbucket.org/rawlings-group/octave-mpctools/src/master/, aufgerufen am 11.03.2020

[3] https://web.casadi.org/, aufgerufen am 11.03.2020

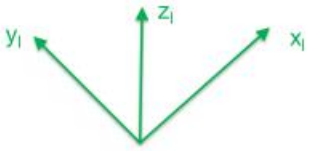
Model of the Drone
crazyflie 2.1

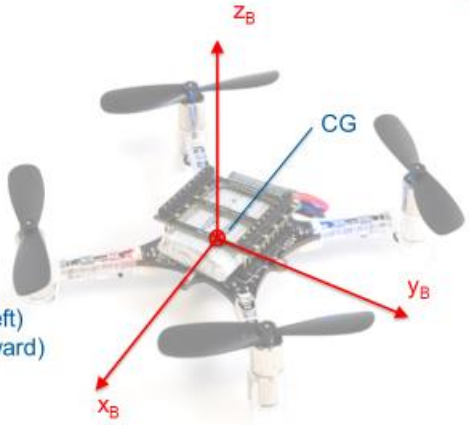
Prof. Dr. Wolfgang Krämer
Technische Hochschule
Ingolstadt, Germany

Model of the Drone
Coordinate Systems

Inertial Coordinate System:
Fixed in space
x_I-y_I-plane is horizontal
z_I-axis is vertical

Body Coordinate System:
Fixed to the body of the drone
Origin in the center of gravity
x_B-axis in forward direction
y_B-axis in lateral direction (to the left)
x_B-axis vertical to x_B-y_B-plane (upward)





For every quantity, it must be defined, to which coordinate system it refers!

2 W. Krämer

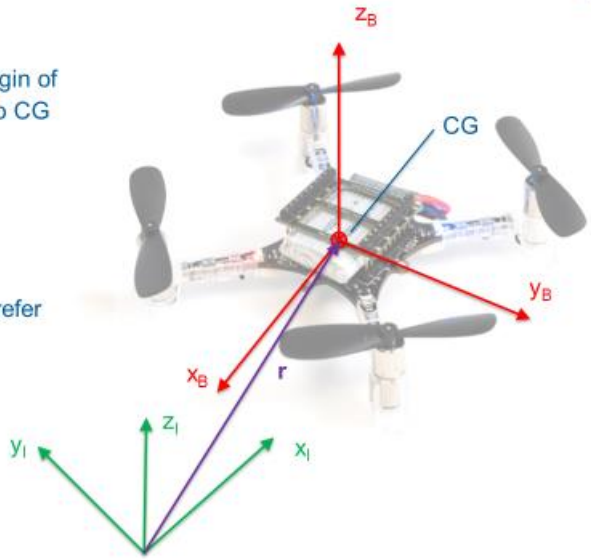
Model of the Drone

Position of the Drone

Position vector from origin of inertial coord. system to CG of the drone:

$$\boldsymbol{r}_I = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

The components of r_I refer to the inertial system.



3 W. Krämer

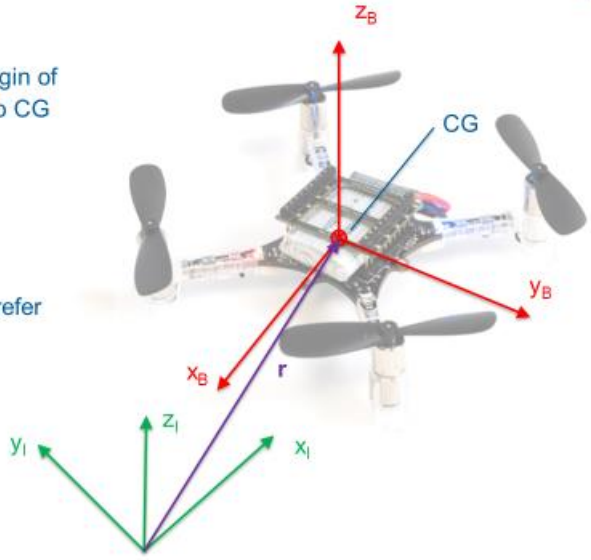
Model of the Drone

Position of the Drone

Position vector from origin of inertial coord. system to CG of the drone:

$$\boldsymbol{r}_I = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

The components of r_I refer to the inertial system.



3 W. Krämer

Model of the Drone

Attitude of the Drone – Cardanian Angles

Rotations:

1) Yaw angle ψ around z_I-axis
2) Pitch angle ϑ around y_1-axis;
   $-\pi/2 \leq \vartheta \leq \pi/2$
3) Roll angle φ around x_2-axis;
   $-\pi \leq \varphi \leq \pi$

The angles are combined in the **attitude vector**

$\boldsymbol{\Phi} = [\varphi \quad \vartheta \quad \psi]^T$

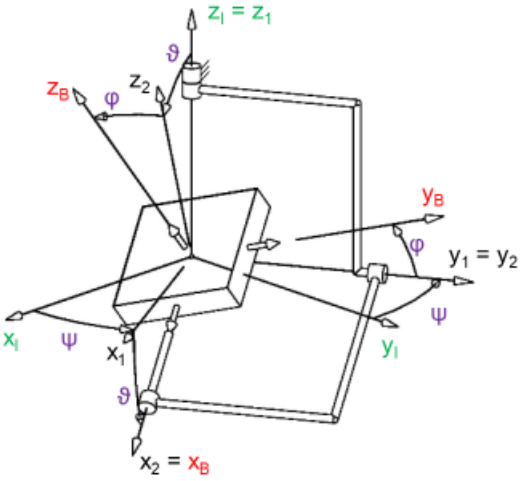


Figure taken from Woernle

Model of the Drone

Attitude of the Drone – Cardanian Angles

Illustration:
Attitude of an airplane

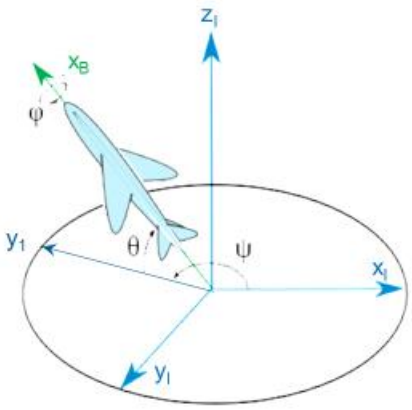


Figure taken from wikipedia

Model of the Drone

Attitude of the Drone – Rotation Matrix

The relation between the components of a vector w, given in the body coord. system and in the inertial system, is described by a linear transformation.

If w_B is the representation of the vector w in the body coord. system and w_I is the representation in the inertial system, then:

$$w_I = R \cdot w_B$$

R is the **rotation matrix**.

It describes the rotation **from** the body **to** the inertial coord. system.

For cardanian angles:

$$R = \begin{bmatrix} \cos\vartheta\cos\psi & \sin\varphi\sin\vartheta\cos\psi - \cos\varphi\sin\psi & \cos\varphi\sin\vartheta\cos\psi + \sin\varphi\sin\psi \\ \cos\vartheta\sin\psi & \sin\varphi\sin\vartheta\sin\psi + \cos\varphi\cos\psi & \cos\varphi\sin\vartheta\sin\psi - \sin\varphi\cos\psi \\ -\sin\vartheta & \sin\varphi\cos\vartheta & \cos\varphi\cos\vartheta \end{bmatrix}$$

R is orthogonal: $R \cdot R^T = I$; $\quad R^{-1} = R^T$

Model of the Drone

Forces acting on the Drone - Thrusts

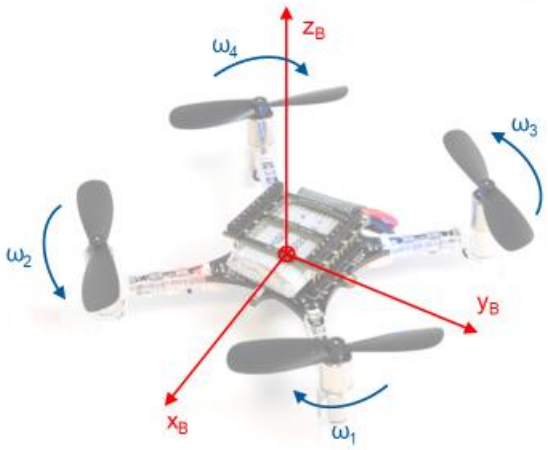
The angular velocities ω_i of the rotors create thrusts

$F_i = k_F \cdot \omega_i^2$

and torques around the z_B-axis (drag)

$T_i = \pm k_T \cdot \omega_i^2$,
$k_F, k_T = const., i = 1, \ldots, 4$

(The accelerations of the rotors are neglected.)



Model of the Drone

Forces acting on the Drone – Total Force and Torque

The total force w.r.t. the body coordinate system is

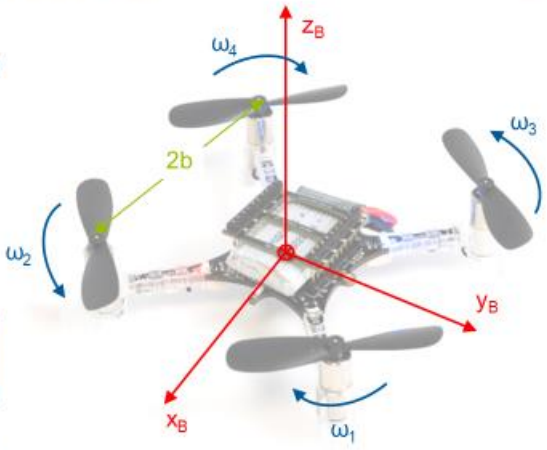
$$\boldsymbol{F}_B = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} \quad \text{with } F = \sum_{i=1}^{4} F_i$$

The total torque is

$$\boldsymbol{T}_B = \begin{bmatrix} b \cdot (F_1 + F_3 - F_2 - F_4) \\ b \cdot (F_3 + F_4 - F_1 - F_2) \\ \kappa \cdot (F_1 + F_4 - F_2 - F_3) \end{bmatrix}$$

with $\kappa = k_T / k_F$

w.r.t the body coord. system.



9 W. Kramer

Model of the Drone

Forces acting on the Drone – Drag Force

Usually, a drag force is taken into account. For reasons of simplicity, this force is considered to be proportional to the velocity.

The drag force in the body coord. system is:

$$\boldsymbol{F}_{D,B} = -\boldsymbol{D} \cdot \boldsymbol{v}_B$$

with the velocity in the body coord. system

$$\boldsymbol{v}_B = \boldsymbol{R}^T \cdot \dot{\boldsymbol{r}}_I$$

and the matrix of drag coeffients

$$\boldsymbol{D} = diag(d_x \quad d_y \quad d_z)$$

So, the drag force w.r.t. the inertial coord. system is

$$\boldsymbol{F}_D = -\boldsymbol{R} \cdot \boldsymbol{D} \cdot \boldsymbol{R}^T \cdot \dot{\boldsymbol{r}}_I$$

10 W. Kramer

Model of the Drone

Equation of Motion – Translational Motion

With the mass m of the drone, the equation of translational motion is

$$m\ddot{\boldsymbol{r}}_I = \boldsymbol{R} \cdot \boldsymbol{F}_B + \boldsymbol{F}_G + \boldsymbol{F}_D$$

The gravitational force $\boldsymbol{F}_G$ is easily represented in the inertial coord. system:

$$\boldsymbol{F}_G = [0 \quad 0 \quad -m \cdot g]^T$$

Model of the Drone

Equation of Motion – Rotation

To describe the rotation of the drone, the vector of angular velocities is used:

$$\boldsymbol{\omega}_B = [\omega_x \quad \omega_y \quad \omega_z]^T$$

It refers to the body coordinate system.

As the x_B-, y_B-, and z_B-axes are the principal axes of the drone, the inertia matrix is

$$\boldsymbol{J} = diag(J_{xx} \quad J_{yy} \quad J_{zz})$$

The equation of rotational motion is (Euler's equ.)

$$\boldsymbol{J} \cdot \dot{\boldsymbol{\omega}}_B = \boldsymbol{T}_B - \boldsymbol{\omega}_B \times \boldsymbol{J} \cdot \boldsymbol{\omega}_B$$

The second term on the right handside considers fictious centrifugal forces.

Model of the Drone

Rates of Change of the Cardanian Angles

In order to calculate the Cardanian angles, a relationship between the angular velocity vector $\boldsymbol{\omega}_B$ and the rate of change $\dot{\boldsymbol{\Phi}} = [\dot{\varphi} \quad \dot{\vartheta} \quad \dot{\psi}]^T$ of the Cardanian angles must be established.

It has to be: $\boldsymbol{\omega} = \dot{\boldsymbol{\varphi}} + \dot{\boldsymbol{\vartheta}} + \dot{\boldsymbol{\psi}}$

Notes: $\dot{\boldsymbol{\varphi}}$, $\dot{\boldsymbol{\vartheta}}$ and $\dot{\boldsymbol{\psi}}$ are not mutually orthogonal.

$\dot{\boldsymbol{\varphi}}$, $\dot{\boldsymbol{\vartheta}}$ and $\dot{\boldsymbol{\psi}}$ must be described in the body coord. system.

From the development of the rotation matrix $\boldsymbol{R}$, it can be derived:

$$\dot{\boldsymbol{\varphi}}_B = [1 \quad 0 \quad 0]^T \cdot \dot{\varphi}$$

$$\dot{\boldsymbol{\vartheta}}_B = [0 \quad \cos\varphi \quad -\sin\varphi]^T \cdot \dot{\vartheta}$$

$$\dot{\boldsymbol{\psi}}_B = [-\sin\vartheta \quad \sin\varphi\cos\vartheta \quad \cos\varphi\cos\vartheta]^T \cdot \dot{\psi}$$

Model of the Drone

Rates of Change of the Cardanian Angles

The equations for the vectors $\dot{\boldsymbol{\varphi}}_B$, $\dot{\boldsymbol{\vartheta}}_B$, and $\dot{\boldsymbol{\psi}}_B$ are added to yield

$$\dot{\boldsymbol{\omega}}_B = \dot{\boldsymbol{\varphi}}_B + \dot{\boldsymbol{\vartheta}}_B + \dot{\boldsymbol{\psi}}_B$$

This linear system of equations can be solved for $\dot{\varphi}$, $\dot{\vartheta}$, and $\dot{\psi}$:

$$\dot{\boldsymbol{\Phi}} = \begin{bmatrix} \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\varphi\tan\vartheta & \cos\varphi\tan\vartheta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi / \cos\vartheta & \cos\varphi / \cos\vartheta \end{bmatrix} \cdot \boldsymbol{\omega}_B$$

The equations of motion and the equ. for $\dot{\boldsymbol{\Phi}}$ completely describe the dynamic behavior of the drone.

Model of the Drone

State Space Description

A reasonable vector of state variables is:

$$\boldsymbol{x} = [\boldsymbol{r}_I^T \quad \dot{\boldsymbol{r}}_I^T \quad \boldsymbol{\Phi}^T \quad \boldsymbol{\omega}_B^T]^T$$

The order of the model is $n = 12$.

There are several possibilities to define the control variables:

- Total force and torque: $\boldsymbol{u} = [F \quad \boldsymbol{T}_B^T]^T$
- Forces of the rotors: $\boldsymbol{u} = [F_1 \quad F_2 \quad F_3 \quad F_4]^T$
- Squared velocities of the rotors: $\boldsymbol{u} = [\omega_1^2 \quad \omega_2^2 \quad \omega_3^2 \quad \omega_4^2]^T$
- Voltages of the motors: $\boldsymbol{u} = [U_1 \quad U_2 \quad U_3 \quad U_4]^T$

With these variables, state equations can easily be set up based on the equations of motion and the equ. for $\dot{\boldsymbol{\Phi}}$.

Model of the Drone

Measurements

The following quantites can be measured:

- Position r_I of the drone by the „Loco Positioning System“, a kind of local GPS (in the inertial coord. system).
- Angular velocities ω_B in 3 directions by a gyroscope mounted on the drone (in the body coord. system).
- Acceleration in 3 directions by an inertial accelerometer mounted on the drone, presumably near the center of gravitiy (in the body coord. system). This acceleration is:

  $$\boldsymbol{a}_B = \boldsymbol{R}^T \cdot \ddot{\boldsymbol{r}}_I - \boldsymbol{\omega}_B \times (\boldsymbol{R}^T \dot{\boldsymbol{r}}_I)$$

  However, the inertial sensor doesn't catch the centrifugal and gravitational accelerations. Therefore, the measured acceleration is:

  $$\boldsymbol{a}_{meas} = \boldsymbol{R}^T \cdot (\ddot{\boldsymbol{r}}_I - [0 \quad 0 \quad -g]^T)$$

- Altitude r_z of the drone by a precisson pressure transducer.

Model of the Drone

State Space Description

If all measurements are used, except of $r_{z'}$ the output vector is

$$\boldsymbol{y} = [\boldsymbol{r}_I^T \quad \boldsymbol{\omega}_B^T \quad \boldsymbol{a}_{meas}^T]^T$$

There are $n_y = 9$ measurements.

Model of the Drone

Structure of the Model

Linearized state space equ. (in steady state $\varphi_{ss} = \vartheta_{ss} = 0$, ψ_{ss} arbitrary):

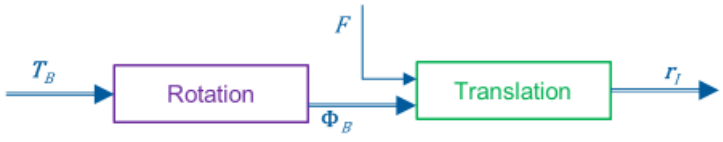
$$\begin{bmatrix} \dot{\boldsymbol{r}}_I \\ \ddot{\boldsymbol{r}}_I \\ \dot{\boldsymbol{\Phi}} \\ \dot{\boldsymbol{\omega}}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{R}_{ss}\boldsymbol{D}\boldsymbol{R}_{ss}^T/m & \boldsymbol{X}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{r}_I \\ \dot{\boldsymbol{r}}_I \\ \boldsymbol{\Phi} \\ \boldsymbol{\omega}_B \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{x}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{J}^{-1} \end{bmatrix} \cdot \begin{bmatrix} F \\ \boldsymbol{T}_B \end{bmatrix}$$

Translational subsystem

Rotational subsystem

with $\boldsymbol{X}_1 \neq \boldsymbol{0}$, $\boldsymbol{x}_m = [0 \quad 0 \quad 1/m]^T$, $\boldsymbol{R}_{ss} = \boldsymbol{R}(\varphi_{ss}, \vartheta_{ss})$

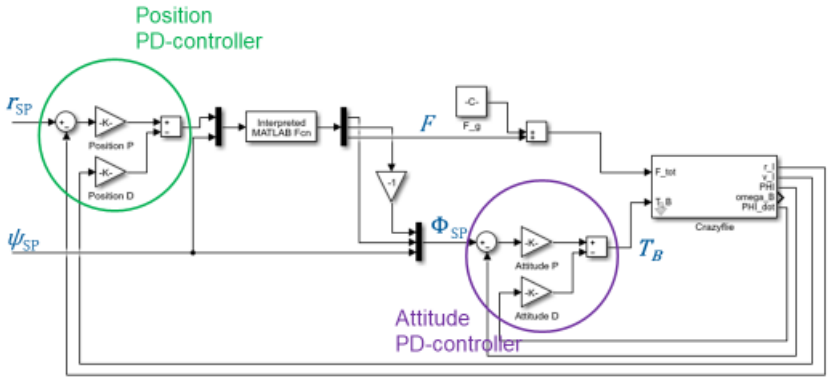
The model shows the structure of a series connection of two subsystems:



Model of the Drone

Application: Simple Control Structure

This structure suggests cascade control as a simple control structure:



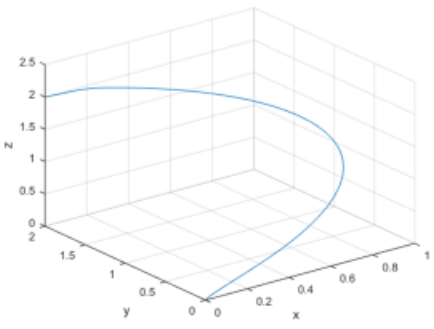
This control system contains some idealizations.

Model of the Drone

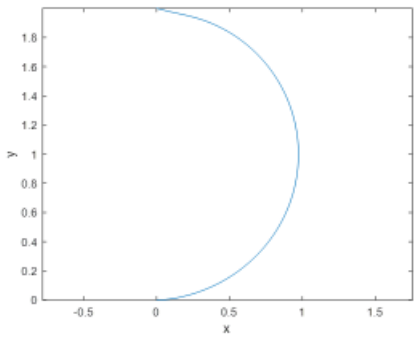
Control Behavior

The desired trajctory of the drone is a semicircle in x-y-direction with radius 1 m in 4 s (t_{Sim} = 6 s). The drone sould turn, so that it flies in forward direction. In addition it should lift by 2 m.

Trajectory in 3-d space



Trajectory in x-y-direction



Model of the Drone

Final Remarks

- In the model, time lags of the rotor dynamics are neglected.
- There are only rough estimates of the parameters of the drone available up to now.
- There might be latency times in the measurements.
- When controlling the drone, there will be computing time delays, presumably.
- Gyroscopic forces and accelerations of the rotors are neglected.
- Aerodynamic effects, like the ground effect or the influence of longitudinal velocities on the rotor forces, are neglected.

Literature:

Alderete, T.S.: Simulator Aero Model Implementation. NASA Ames Research Center, Moffet Field, California, 1995.

Status Report 1 of Drone Project: State Feedback

Some simulations of the drone with simple controllers have been carried out. The model of the drone described in the presentation1 "Model of the Drone" is used; it will simply be called "presentation" in the sequel. So, the state variables are:

- The positions (position vector $\boldsymbol{r}_I$) w.r.t. inertial coord. system.
- The velocities (velocity vector $\boldsymbol{v}_I = \dot{\boldsymbol{r}}_I$) w.r.t. inertial coord. system.
- The attitude angles (vector $\boldsymbol{\Phi}$).
- The angular velocities (vector $\boldsymbol{\omega}$) w.r.t. body coord. system.

Each of these vectors has three components; the order of the drone model is therefore 12.

The following four control variables (inputs) are used:

- The thrust or total force of the rotors F.
- The torques (vector $\boldsymbol{T}_B$) w.r.t. body coord. system.

Approximate values for the parameters of the drone are used, based on the literature [1]. [2]. Small drag force coefficients are selected arbitrarily ("linear damping").

For testing of simple controllers, a moderate test maneuver is used: The trajectory of the drone should describe a semicircle with a radius of 1 m over the x-y-plane; the drone should run through this semicircle within 4 s and then stop. It should follow this path in forward direction, i.e. it has to yaw by 180°. In addition, its altitude should be increased by 2 m within the first 2 s. The total simulation time is 5 s. No careful tuning of the controllers is done yet; the goal of the simulations is to test the model of the drone and to check, whether simple controllers work at all.

1 Simulink Model

A Simulink model of the drone with simple controllers is set up. This model is shown on the last pages of the presentation. PD-controllers for the positions and the angles in a partial cascade structure are used. The controllers for the *x*- and *y*-positions determine reference values for the pitch and roll angle. A transformation (rotation) is used, to assign the command variables of the

[1] The model was presented in the Rawlings group at May 19, 2020

position controllers to the angles. This transformation depends on the reference value of the yaw angle. The *z*-position controller determines the force F and the attitude (angle) controllers determine the torques $\boldsymbol{T}_B$.

The presentation also shows a simulation result. Obviously, the simple controllers work well with the model of the drone, but are not quite realistic; they use the velocities of the drone, which are not measured, and work in continuous time, for example.

2 Simulations in Matlab with MPCtools

The model of the drone is implemented in Matlab using MPCtools. A time step of 0.01 s is used in all Matlab simulations. A LQR based on the linearized model (c.f. CSTR example in MPCtools) without state estimation is used for control. The linearization is done for $\boldsymbol{x} = \boldsymbol{0}$ (crazyflie_1.m)2.

This simple controller only works for small yaw angles ($|\psi| < 30°$, approximately). Therefore, the test maneuver cannot be performed. It can only be performed in a simplified way, in which the drone doesn't yaw, i.e. it doesn't rotate around its vertical axis. So, it doesn't follow the trajectory in its forward direction.

Gain scheduling is used, so that the controller also works for larger yaw angles. Three versions of gain scheduling are tested:

1. Linearization and LQR design are performed at each time step (c.f. gainscheduling example in MPCtools) based on the desired values of the positions and the yaw angle; the yaw angle is most important. All other state variables are set to zero for linearization (crazyflie_2.m). This gain scheduling leads to high computational load of the control computer, see table below. Therefore, other versions are tested.

2. Linearization and LQR design are performed in advance, before the simulation of the control system, for sampling values of the yaw angle in the range $-\pi \leq \psi \leq \pi$ with an increment of π/10, and the feedback matrices are stored. During the simulation the feedback matrix is determined in each time step by linear interpolation based on the stored matrices in dependence of the desired yaw angle.

   This also requires some computational load for the control computer and also some storage requirements for the feedback matrices. These requirements could be

[2] The m-files with the corresponding simulation models are mentioned in parenthesis.

reduced, if only the matrix elements, which change considerably with the yaw angle, are interpolated. In addition, some matrix elements are very small (zero?).

Question: Is something known about the structure of LQR feedback matrices, i.e. which of their elements are zero?

3. Feedback matrices are calculated in advance in the same way as in version 2, but the interpolation is omitted. The feedback matrix for the sampling value, which is closest to the current desired yaw angle is used (crazyflie_4.m); therefore, the feedback matrix is switched.

   The storage requirements of this version can also be reduced as described for version 2.

All of these gain scheduling controllers work well for the test maneuver. With version 2, there are very small peaks in the control variables, if the sampling interval of the yaw angle changes. With version 3, there are considerable peaks, when the feedback matrix is switched. These peaks in the control variables lead to small changes or some "restlessness" of the attitude angles. These effects could be reduced by using a smaller yaw angle increment for the LQR designs.

The following table shows the run times of the control system. It gives an impression of the computational loads required for the different versions. The versions described in the next section are included.

Model, controller	*Total run time [s]*	*Time for linearizarion [s]*	*Time for LQR calculation [s]*
crazyflie_1	0.92	0.10	0.22
crazyflie_2	8.50	3.22	4.47
crazyflie_3	1.54	0.25	0.45
crazyflie_4	1.42	0.24	0.42
crazyflie_10 w/o inertial quantities	0.91	0.10	0.22
crazyflie_10 w/ inertial quantities	0.99	0.12	0.25
crazyflie_11	0.92	0.10	0.22

Only crazyflie_2 requires considerably more time than the other versions. This is due to the linearization and LQR calculation in every time step. The difference between crazyflie_3 and crazyflie_4 is the interpolation; it seems that this doesn't require much computational effort. (The times for linearization and LQR calculation for these two versions should be equal. Reason for the small differences?)

3 Modifications of the Model

The gain scheduling is necessary, because the applied toques, i.e. the control variables, and the controlled states, i.e. the positions, are not quite consistent in the model of the drone: The torques are defined w.r.t. the axes of the drone (body coord. system) whereas the positions are described in the inertial system. This leads to the encountered problem with constant feedback, if the yaw angle becomes too large.

To overcome this problem, the translational velocities and the positions are now defined w.r.t. the body coord. system. This requires only a small change in the model; the main difference to the first model is, that the centrifugal force must be taken into account (crazyflie_10).

In this model, the velocities and positions w.r.t. the inertial system are also calculated for reasons of evaluation and comparison of the control system.

With this modified model, a constant feedback matrix, calculated on basis of the linearized model for $\boldsymbol{x} = \mathbf{0}$, is sufficient for execution of the test maneuver without lift. Of course, the reference trajectory must also be specified w.r.t. the body coord. system. This is easy for the test maneuver, but could be cumbersome for complex trajectories. If a trajectory described w.r.t. the inertial system is to be transformed to the body coord. system, the current attitude angles shouldn't be used, because this would lead to additional feedback and could cause instability of the control system. So reasonable values for these angles must somehow be determined based on the required trajectory.

The use of the modified model (and constant feedback) leads to a drawback: The drone loses height during the maneuver. The reason for this is quite obvious, as its longitudinal axis (x-axis) points slightly downward in order to generate a longitudinal driving force. In order to overcome this disadvantage, another modification of the model is made: The vertical position of the drone (z-position) is described w.r.t. the inertial system (crazyflie_11.m). This is an ad hoc solution and should be tested in more detail.

Another disadvantage of the modifications is, that the drone seem to follow the desired trajectory with less accuracy than with the gain scheduling control. The reasons for this might be, that the positions w.r.t the inertial system are not controlled and that these positions are calculated by simple Euler integration of the velocities.

It could also be tried to change the inputs (torques) of the drone, so that they are more consistent with the controlled positions. But this would make it difficult to formulate constraints for the inputs, which will be considered in further development.

4 Next Steps

The next steps in this project could be:

- Add state estimation.
- Make the controllers really predictive.
- Take constraints of the control variables and states into account.
- Use nonlinear control.
- Determine, which models are best suited for MPC.

References

[4] https://www.bitcraze.io/products/crazyflie-2-1/, April 22, 2020.

[5] Landry, B.: Planning and Control for Quadrotor Flight through Cluttered Environments. B.S. MIT, 2014.

Status Report 2 of Drone Project: State Estimation

The focus of the investigations during the last weeks was state estimation. For these investigations, the model of the drone described in the presentation3 “Model of the Drone” is used; it will simply be called “presentation” in the sequel. All investigations are based on simulations.

For testing, the same maneuver as in the first report is used: The trajectory of the drone should describe a semicircle with a radius of 1 m over the x-y-plane; the drone should run through this semicircle within 4 s and then stop. It should follow this path in forward direction, i.e. it has to yaw by 180°. In addition, its altitude should be increased by 2 m within the first 2 s. The total simulation time is 5 s. For feedback, the LQR with gain scheduling is used (crazyflie_2 in report 1). Unless otherwise noted, the time step is $\Delta t = 0.01$ s.

The following state estimation schemes were implemented and tested:

1. Linear steady state Kalman filter with “gain scheduling” (KF).
2. Extended Kalman filter (EKF).
3. Moving horizon estimation (MHE).
4. Madgwick filter (MF).

All estimators are tested in open loop at first, i.e. the state estimates are not used for feedback; ideal state feedback is used in this case. In a second step, the state estimates are used for feedback, i.e. the estimators operate in closed loop.

In order to assess the impact of disturbances, normally distributed measurement noises with the following standard deviations are introduced in the simulation model:

1. Position measurements: $s_p = 0.05\ \mathrm{m}$; according to bitcraze4, the accuracy of the positioning system is about 10 cm.
2. Measurement of angular velocities: $s_\omega = 0.0056\ \mathrm{rad/s}$, according to the data sheet5 of the sensor BMI088.
3. Acceleration measurements in x- and y-direction: $s_{a,xy} = 0.026\ \mathrm{m/s^2}$, acceleration measurement in z-direction: $s_{a,z} = 0.029\ \mathrm{m/s^2}$, according to the data sheet of the BMI088.

[3] The model was presented in the Rawlings group at May 19, 2020

[4] https://www.bitcraze.io/products/loco-positioning-system/ (6-11-20)

[5] https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi088.html (6-11-20)

These are maximum values for the standard deviations: The inaccuracies of the position measurements need not totally be due to noise; there could also be offsets or slowly changing parts. The standard deviations of the inertial measurements (BMI088) can be reduced by choosing a smaller bandwidth.

In addition, normally distributed state noise with arbitrarily chosen standard deviation s_w is introduced.

1 Linear Steady State Kalman Filter with "Gain Scheduling"

This filter is based on the same idea as the LQR described in report 1: The model of the drone is linearized at each time step for the current reference variables and a steady state KF is redesigned. The linearization is performed for steady state, i.e. the roll and pitch angles are $\varphi_{ss} = \vartheta_{ss} = 0$, c.f. presentation p. 18. So in the linearizations, only different values of the yaw angle reference ψ_{ref} are used.

This approach faces a serious problem: The linearized system is not observable (also not detectable). The reason is, that the submatrix $\boldsymbol{X}_1$ in the linearized model is singular (rank 2), c.f. presentation p. 18. This submatrix couples the rotational into the translational subsystem; it is the Jacobian

$$\boldsymbol{X}_1 = \partial \boldsymbol{F}_I / \partial \boldsymbol{\Phi},$$

where $\boldsymbol{F}_I = \boldsymbol{R}\boldsymbol{F}_B$ is the force vector acting on the drone in the inertial coordinate system, and $\boldsymbol{\Phi}$ is the attitude vector, c.f. presentation pp. 5, 9, and 11.

This Jacobian has the null vector

$$\boldsymbol{n} = \begin{bmatrix} \sin\vartheta \\ -\cos\vartheta \tan\varphi \\ 1 \end{bmatrix}.$$

In steady state, it is $\boldsymbol{n} = [0 \quad 0 \quad 1]^T$. Obviously, the yaw angle ψ does not influence the translational subsystem, i.e. it is not reflected in the position measurements, and is therefore unobservable – according to the system description obtained by linearization in steady state.

To overcome this problem, two approaches are tested:

a) Calculate the yaw angle based on the measurements of the angular velocities, i.e. integrate $\dot{\psi}$, as given on p. 14 of the presentation. Simple Euler forward integration is used for this task. Then the calculated yaw angle is considered an additional measurement value.

b) Use the reference value ψ_{ref} as additional measurement value. If the feedback control works well, $\psi_{ref} \approx \psi$.

For the system noise standard deviation, $s_w = 0.01$ is used.

In open loop the KF gives reasonable results. It makes almost no difference, whether the integrated or the reference value is used as additional measurement value for ψ. However the estimated vertical velocity $\hat{\dot{r}}_z$ has a considerable steady state offset; this should be due to the linearization at $\varphi_{ss} = \vartheta_{ss} = 0$ (steady state).

If the KF is operated in closed loop, the control loop is stable. However; the closed loop behavior deteriorates considerably in comparison with the ideal case, where the states are fed back: The projection of the trajectory in the x-y-plane doesn't look like a semicircle anymore, and the angles and velocities flutter strongly. The closed loop behavior can be improved a little bit by retuning the LQR. If the reference value ψ_{ref} is used as additional measurement value, the closed loop behavior is worse than with the integrated value.

With standard deviation $s_w = 0.001$, the behavior becomes much more quiet, but there occur large steady state errors in the vertical position of the drone.

Larger time steps, e.g. $\Delta t = 0.05$ s, are possible: the closed loop is still stable, but there also occur large steady state errors in the vertical position of the drone.

2 Extended Kalman Filter

In open loop, the EKF delivers good estimates with standard deviation $s_w = 0.01$. However, using the estimates for feedback, also results in much poorer, but stable control behavior, not much better than with the KF. A standard deviation of $s_w = 0.001$ for the state noise results in a much better and smoother behavior without other disadvantages.

It is possible to use larger time steps; e.g. $\Delta t = 0.05$ s only results in a little rougher transients. It is also possible to neglect the acceleration measurements; this also deteriorates the control behavior only slightly.

The reason for the large control errors is the measurement noise. When applying feedback of state estimates, it should be checked, whether the bandwidth of the inertial sensor can be reduced in order to reduce the measurement noise. It should also be investigated, whether a retuning of the LQR leads to better results

3 Moving Horizon Estimation

When applying MHE6, there is the problem, that the system has a direct transmission from the control variable total force to the acceleration measurement – at least there is no straight forward way to deal with such systems in mpctools. To solve this problem, a time lag for the build up of the force is introduced, which is quite realistic ($T_1 = 0.025\ s$ arbitrarily chosen). No lag is used for the torques (which is not realistic) to keep the system order low.

For open loop estimation, it is difficult to find parameters, for which the nonlinear solver is successful. The time step of $\Delta t = 0.01\ \mathrm{s}$, which was usually used up to now, is too small; $\Delta t = 0.05\ s$ should be used. The horizon must not be too small, it should be 20 or more. In addition, the weighting of the prior strongly influences the success of the solver. The estimates obtained with parameters, for which the solver works, are of limited quality. They are not better than with the EKF.

Using the state estimates for feedback7, results in quite poor control behavior, not better as with the EKF. The reason for this is also the measurement noise. If no noise is injected in the simulation, the state estimates and the closed loop behavior are fine.

Before thinking about using MHE for control of the real drone, more investigations should be made, if MHE could be taken into consideration for this task at all.

4 Madgwick Filter

According to bitcraze8, a MF [1] is used in the standard control software of the drone to estimate the attitude based on the measurements of acceleration and angular velocity. It uses quaternions to describe the attitude; this has the advantage that no trigonometric functions are necessary and so the computations are cheap. In order to compare the results of the MF with the true states, the quaternions are converted to Cardanian angles in the current investigations.

The MF basically integrates the measured angular velocities by Euler forward integration. This corresponds to integrating the derivatives given on p. 14 of the presentation. The result is corrected by using the acceleration measurements. The basic idea for this is, that the direction of

[6] Nmheexample of the mpctools examples is used as a blue print.

[7] Many thanks to Koty McAllister for his help to get this running.

[8] https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor_to_control/ (04/20/2020)

the acceleration measurements in the inertial coordinate system is known: It is the vertical direction – of course this is true only in steady state. So the correction is done in such a way that the direction of the measured acceleration aligns better with the known direction. For this an "optimization problem" is formulated, which actually is searching the solution of a set of nonlinear equations9. A Newton step is used to improve the attitude estimation. Of course the MF is not a filter in the sense of a KF, as it doesn't use a dynamic model of the system and as it is not possible to consider noise in any way.

The integration of the angular velocity measurements works well – without measurement noise or errors. But it was not possible to get reasonable results, when the "correction" was put into operation, even after fixing some errors in [1]. This is quite strange, because a simple C-code is given in [1] (and also by bitcraze), which was transferred into a Matlab function.

If the MF could be made working reasonably, it could be combined with a KF or EKF: The MF would estimate the attitude of the drone and the KF the position and velocity. This could probably reduce the computational effort compared to a full order KF / EKF. In [1] the extension of the MF by another KF is proposed to take account of measurement offsets of the angular velocity sensors.

5 Summary

Based on the simulation results it seems, that the EKF10 is well suited for state estimation. The MHE has some problems; there are also doubts, that the MHE can be calculated in real time. So it is questionable whether it is worthwhile trying to improve the MHE.

When using the state estimates for feedback, the magnitudes of the measurement noises are very important. It should be tried to get realistic values for them, which are hopefully smaller than those used in the simulation, and to reduce them as far as possible, e.g. by reducing the bandwidths of the sensors.

It should be possible to reduce the computational load of the KF / EKF by exploiting the structure of the linearized system, as the matrices of this system contain elements or blocks, which are always zero. Another question is, whether quaternions should be used to describe the attitude of the drone. This would possibly reduce the computational load, because no

[9] There are inaccuracies and errors in [1].

[10] bitcraze also offers code of an EKF; this was not looked at in the current investigations.

trigonometric functions need to be used. But quaternions are much more abstract than Cardanian angles, which is a serious disadvantage, if the drone is to be used by students in the control lab.

References

[1] Madgwick, S.O.H: An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Report x-io and University of Bristol (UK) 25 (2010): 113-118.

Status Report 3 of Drone Project: Structure of LQR Feedback Matrix, Constraints, and Madgwick Filter Continued

In this report, some short results are summarized:

1. The structure of the LQR feedback matrix.
2. Consequences of constraints of the control variables.
3. Additional results with the Madgwick filter.

The investigations are based on the model of the drone described in the presentation11 "Model of the Drone"; it will simply be called "presentation" in the sequel. This model uses the state vector

$$\boldsymbol{x} = [r_x \quad r_y \quad r_z \quad \dot{r}_x \quad \dot{r}_y \quad \dot{r}_z \quad \varphi \quad \vartheta \quad \psi \quad \omega_x \quad \omega_y \quad \omega_z].^T$$

The first three variables describe the position of the drone (w.r.t. inertial coordinate system), the next three variables are the corresponding velocities. Variables 7 – 9 are the roll, pitch and yaw angles, followed by the angular velocities (w.r.t. body coordinate system).

For the calculation of the LQR considered in the following sections, the weighting matrices

$$\boldsymbol{Q} = \mathrm{diag}[10^3 \quad 10^3 \quad 10^3 \quad 0 \quad 0 \quad 0 \quad 100 \quad 100 \quad 100 \quad 1 \quad 1 \quad 1]$$

$$\boldsymbol{R} = \boldsymbol{I}$$

are used unless otherwise noted.

1 Structure of the LQR Feedback Matrix

In the first report, it is already mentioned, that many elements of the feedback matrix are small. This is looked at in more detail in this section. The control variables are

$$\boldsymbol{u} = [F \quad T_x \quad T_y \quad T_z]^T,$$

where F is the total thrust, and T_i is the torque around the i-axis, $i \in \{x, y, z\}$.

As the linearized model depends on the yaw angle, the feedback matrix also depends on ψ – this is the reason for the gain scheduling. The feedback matrix has the following structure:

[11] The model was presented in the Rawlings group at May 19, 2020

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & X & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & 0 & X & X & 0 & X & X & 0 & X & X & 0 \\ X & X & 0 & X & X & 0 & X & X & 0 & X & X & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & X \end{bmatrix}.$$

X denotes elements, which are not for all values of the yaw angle ψ zero (not always zero)

Computational details: When the feedback matrix is calculated by Matlab, the zero-elements are not exactly zero. But it can be seen, that there are two clearly distinguishable groups of elements: Large elements and small elements. E.g. for $\psi = \pi/4$, the large elements are in the range $10^{-7} < |k_{i,j}| < 10^{2}$ (actually, only $|k_{3,10}|$ and $|k_{2,11}|$ are of the order 10^{-7}, the other large elements are at least of order 10^{-5}), while for the small elements $|k_{i,j}| < 10^{-13}$. This gives reason for the assumption that the small elements should be zero.

The same is true for the elements of the solution $\boldsymbol{P}$ of the steady state Riccati equation. There are also clearly distinguishable groups of large and small elements (the orders of magnitude are a little different than for $\boldsymbol{K}$). If the small elements are set to zero, the modified $\boldsymbol{P}$ still solves the Riccati equation with the same accuracy, i.e. the largest elements of the right hand side of the Riccati equation are of the same order of magnitude as for the unmodified solution (10^{-10}). If the feedback matrix is calculated on the basis of the modified $\boldsymbol{P}$-matrix, the above given zero elements are obtained exactly.

This leads to some conclusions:

- If the linearization is done for $\psi = 0$, the elements $k_{2,1}$, $k_{3,2}$, $k_{2,4}$, $k_{3,5}$, $k_{3,10}$, and $k_{2,11}$ are also zero. This shows, that the LQR does the same as the simple controller shown in the presentation: PD-control of the positions and the Cardanian angles – of course without cascade structure. It is possible to transfer the LQR to the PD-controller structure (and vice versa).
- If the gain scheduling is done by interpolation or selection of the feedback gains as described in the first report, only the elements marked X must be interpolated or selected. This reduces the computing load considerably.
- In the calculation of the control variables, only the X-gains must be considered. This leads to further reduction of the computing load.

2 Consequences of Constraints of the Control Variables

A simulation study is performed to assess the consequences of the constrained manipulated variables and test some simple measures to alleviate theses consequences. In this study, the forces of the four rotors are used as control variables, so

$$\boldsymbol{u} = [F_1 \quad F_2 \quad F_3 \quad F_4]^T$$

The constraints are $0 \leq F_i \leq 0.12\ \mathrm{N},\ i = 1, \dots, 4$, according to [1].

For feedback, the LQR with gain scheduling is used (crazyflie_2 in the first report). The time step is $\Delta t = 0.01$ s. No state estimation is used. It is also not tried to apply nonlinear control methods, which can take the constraints of the control variables into account explicitly.

For testing, the maneuver described in the first report is slightly modified: The trajectory of the drone should still describe a semicircle with a radius of 1 m over the x-y-plane; the drone should run through this semicircle within 4 s and then stop. It should follow this path in forward direction, i.e. it has to yaw by 180°. But now its altitude should be increased with velocity v_z during the first 2 s; actually v_z is the rate of change of the setpoint of the vertical position. The total simulation time is 5 s.

2.1 Simple Limitation of the Forces

At first, forces, which exceed their boundary values, are simply set to the corresponding boundary value, i.e. they are clipped. It turns out, that the control loop becomes unstable, if the vertical velocity v_z is chosen too large. One reason of this problem is that the torques are severely changed by the force clipping; if all forces exceed their limits, no torques can be generated any more. This gives reason to trying to retune the LQR in order to attenuate the consequences of the limiting of the forces or to increase the stability limit w.r.t. v_z.

The results of the trials are shown in the following table. In the first columns, the design parameters used, i.e. the weighting factors are given. The last column shows the largest velocity $v_{\mathrm{z,max}}$, for which the control loop is stable. This stability limit is determined with a resolution of 0.05 m/s. If the value $v_{z,\mathrm{max}}$ is chosen, the overall control behavior is quite poor; the projection of the trajectory in the x-y-plane has only limited resemblance to a semicircle. The first row describes the standard parameter set as given above (printed in grey).

Weight of vert. position	Weights of angles		Weights of controls	Stability limit [m/s]
$q_{3,3}$	$q_{7,7} = q_{8,8}$	$q_{9,9}$	$\boldsymbol{R}$	$v_{z,\max}$
1000	100	100	$\boldsymbol{I}$	1.70
100	100	100	$\boldsymbol{I}$	2.35
1000	100	1000	$\boldsymbol{I}$	1.55
1000	1000	1000	$\boldsymbol{I}$	1.50
100	100	1000	$\boldsymbol{I}$	2.15
100	1000	1000	$\boldsymbol{I}$	2.25
1000	100	100	$\boldsymbol{10 \cdot I}$	2.40

Obviously, reducing the weighting factor $q_{3,3}$ of the vertical position increases the stability limit as the forces are limited not so often. It is remarkable that increasing the weights of the angles doesn't help at all. The measures, which increase the stability limit, including increasing the weights on the forces, deteriorate the loop behavior, if the constraints of the forces are not active, i.e. the behavior becomes slower and the control deviations increase.

2.2 Limitation of the Forces by a Constant Fector

This limitation reduces all forces by the same factor. If at least one force exceeds one of its boundaries, the factor is determined such that the largest (smallest) force just reaches its upper (lower) boundary. Then all forces are multiplied by this factor. With this method, the direction of the control vector $\boldsymbol{u}$ is not changed. So, the ratios of the forces are not changed; it is still possible to exert torques, when forces are limited.

The results obtained with this limiting by a constant factor are shown in the following table. For reasons of comparison, the results of the clipping are repeated.

For all tested parameter sets, the results obtained with the limiting by a constant factor are superior to those of the clipping. The stability limit is increased by 54 % in average. Increasing the weighting factors of the angles can also have a positive influence on the "stability region", if the weights of all angles are changed.

Weight of vert. pos.	Weights of angles		Weights of controls	Clipping: Stability limit [m/s]	Limitation by factor: Stability limit [m/s]
$q_{3,3}$	$q_{7,7} = q_{8,8}$	$q_{9,9}$	$\boldsymbol{R}$	$v_{z,\max}$	$v_{z,\max}$
1000	100	100	$\boldsymbol{I}$	1.70	2.25
100	100	100	$\boldsymbol{I}$	2.35	3.25
1000	100	1000	$\boldsymbol{I}$	1.55	2.05
1000	1000	1000	$\boldsymbol{I}$	1.50	3.05
100	100	1000	$\boldsymbol{I}$	2.15	3.05
100	1000	1000	$\boldsymbol{I}$	2.25	4.50
1000	100	100	$\boldsymbol{10 \cdot I}$	2.40	3.25

More parameter sets could be tested, of course. In particular, the weighting factors of the translational and rotational velocities might be increased, but this would be a tedious task.

Another possibility to avoid stability problems would be an elaborate trajectory planning, which avoids frequent and severe limitations of the control variables.

3 Additional Results with the Madgwick Filter

This filter [2] and some results obtained by it are already described in the second report. Here are some additional remarks:

- The estimated values of the MF are not bad, if a small value is chosen for the coefficient β, which determines the degree of correction based on the acceleration measurements, e.g. $\beta = 0.05$. Nevertheless, the "correction" leads to estimation errors, if the drone is not in steady state, i.e. if an important assumption of the filter derivation is not fulfilled. But when steady state is reached, the estimates converge (slowly) towards the true values.
- One could think about using the correction only in steady state – or in a state, which doesn't cause errors – and switching it off otherwise. But then it would be necessary to identify steady state based on the measurements of the angular velocities and accelerations.

- If measurement and state noises are introduced[12], the estimates deteriorate to some degree, but not dramatically. The high frequency components of the noises are attenuated, as the filter basically integrates the measurements, but the low frequency components lead to some "floating" of the estimates. This might be reduced by adding a Kalman filter, which is proposed in [2] in order to avoid estimation errors due to bias of the gyroscopic measurements.

References

[6] PWM to Thrust: https://www.bitcraze.iomisc:investigations:thrust, April 16, 2020.

[7] Madgwick, S.O.H: An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Report x-io and University of Bristol (UK) 25 (2010): 113-118.

[12] The same noise standard deviations as described in the second report are used.

Note on Drone Project: First Experimental Results

On Aug.6, 2020 some measurements with the real drone crazylie 2.1 were taken in the undergrad lab at UCSB. Some results of the stationary measurements are summarized in the sequel. During these measurements, the drone was sitting at a fixed position without any movement. So it should be possible to determine the standard deviations of the measurement noise out of the collected data.

Standard Deviations of Measurement Noise

The following results are based on the data files stationaryhigh and stationarylow, containing 833 and 774 samples of every measured quantity, respectively. Unless otherwise noted, the standard deviations determined from the different measurements coincide.

Acceleration measurements: The standard deviations are independent of the direction (axis) of the measurement: $s_a = 0.0011$ m/s^2.

Angular velocity measurements: There are differences depending on the measurement axis:

$s_{\omega,x} = 0.13$ deg/s

$s_{\omega,y} = 0.18$ deg/s

$s_{\omega,z} = 0.11$ deg/s.

These standard deviations are much smaller than those calculated from the specifications in the data sheet, which are $s_{a,\text{spec}} = 0.026$ m/s^2, $s_{\omega,\text{spec}} = 0.32$ deg/s. The reason for the differences might be that the measured values are already filtered or that the bandwidths of the sensors are reduced.

For the position of the drone, only the results of the onboard Kalman Filter could be measured. If the drone is located well inside the cuboid defined by the nodes of the positioning system, the corresponding standard deviations are for all three directions approximately:

$s_{pos,\text{Kalman}} = 0.01$ m.

For the measurement “stationarylow”, the drone was placed on the floor, so that is was located under the lower nodes of the positioning system ($z \approx -0.1$ m). Then the standard deviation is $s_{z,\text{Kalman,low}} = 0.016$ m. So it can be recommended, that the drone should be operated within the cuboid of the nodes.

To get an idea of the standard deviations of the unfiltered position measurements, simulations of steady state behavior with an Extended Kalman Filter were run. In the simulations, the standard deviations of the position measurements were adjusted so that the results of the Kalman Filter coincid approximately with the measured $s_{pos,\mathrm{Kalman}}$. The process noise was set to zero in the simulations. The result is that the standard deviation of the position measurements is

$s_{pos} \approx 0.025$ m.

This result is a little uncertain, as the tuning of the onboard Kalman Filter is not known. However it is a quarter of the accuracy of the position measurements of 10 cm mentioned by bitcraze and therefore reasonable.

Consequences

The measurement noise standard deviations determined from the measurements are smaller than those used in the investigations described in report 2, which are based on data sheets. If the new values are used, the results of the Extended Kalman Filter are much better, and the state estimates can be used for feedback very well.

Mass of the Drone

During the lab session, the drone was weighted. Its mass is 33 g. This is more than the value used in the simulations performed (27 g). The difference should be caused by the receiver of the positioning system, which was not taken into account up to now.

Path Planning for the Drone crazyflie 2.1

Prof. Dr. Wolfgang Krämer
Technische Hochschule
Ingolstadt, Germany

Contents

Overview of Previous Work

- Control of the Drone
- State Estimation
- Practical Experiences

Path Planning Using MPC (Trajectory Planning)

- Motivation for Path Planning
- Simple Path Planning
- Trial on Time Optimal Path Planning

Overview of Previous Work

Control of the Drone (Simulation Studies)

- Controller: LQR with gain scheduling. Feedback gains are adjusted to yaw angle setpoint.
- LQR has same structure as PD-controllers for positions and angles.
- LQR works well for moderate desired trajectories.
- Control loop with LQR can become unstable due to limitations of the control variables (forces of the rotors), if very fast position changes are demanded.

State Estimation (Simulation Studies)

Methods: Kalman Filter with gain scheduliing
Extended Kalman Filter
Moving Horizon Estimation

Results:

- EKF is superior to KF.
- MHE is not superior to EKF but difficult to apply and computationally most expensive.
- Using estimates for feedback deteriorates control behavior.
  → Determine magnitudes of measurement noise of real drone!

Practical Experiences

- Onboard controllers available: Cascaded PID and state feedback.
- PID control works satisfactory (see following pages);
  sample time of PID controllers: 2 and 4 ms.
- Setpoints can be sent to the drone with a sample time of 100 ms.
- Measurements can be taken with a sample time of 10 ms.
- Presumption: It would require a huge effort to develop other onboard controllers.
- Presumption: Data transfer might be too slow to run control algorithms on a PC.

Path Planning Using MPC (Trajectory Planning)

Motivation

- Onboard controllers work satisfactory.
- It would be time-consuming to program other algorithms for onboard control.
- Boundaries can be taken into account in MPC.
- We want to do something with MPC.
- It is not clear, whether MPC could be run on the onboard processor in real time.
- We want to develop something, which can be used in the (undergrad) lab.

Path Planning for the Drone

Path Planning - Motivation

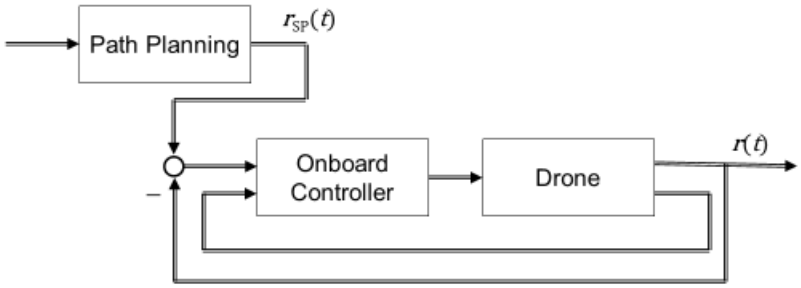
Benefits of path planning with MPC for the lab:

- MPC is used.
- Complexity of the problems can be scaled.
- No real time requirements; path planning can be done offline in advance.
- Many illustrative problems can be set up.
- No difficult programming is necessaty: Octave or Matlab with mpctools can be used.

Path Planning for the Drone

Path Planning - Structure

Structure of the Control System [1]. [2]



r is the position vector of the drone, c.f. next slide.

In addition to r, the yaw angle could also be commanded.

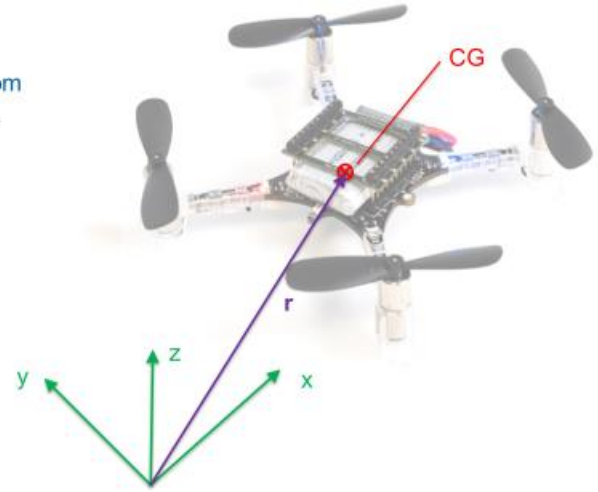
Path planning can be done offline (not in real time).

Position Vector

Position of the drone described by vector from origin of inertial coord. system to CG:

$$\boldsymbol{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

The components of $\boldsymbol{r}$ refer to the inertial coordinate system.



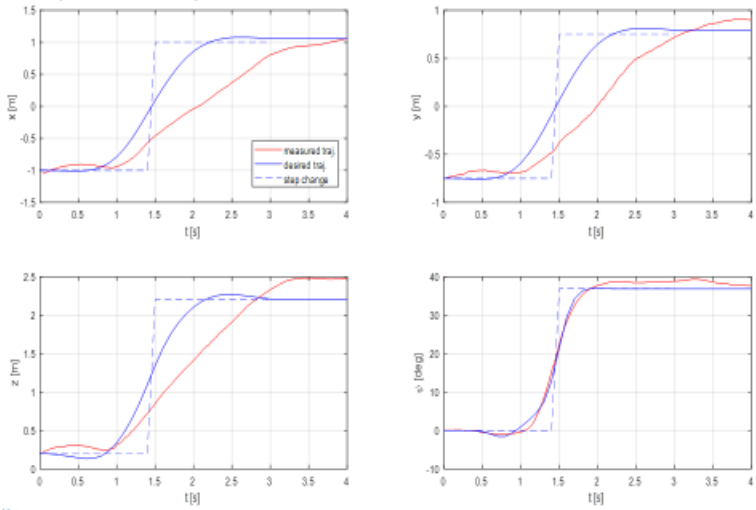
Simple Path Planning

Experiment 1: Point to Point

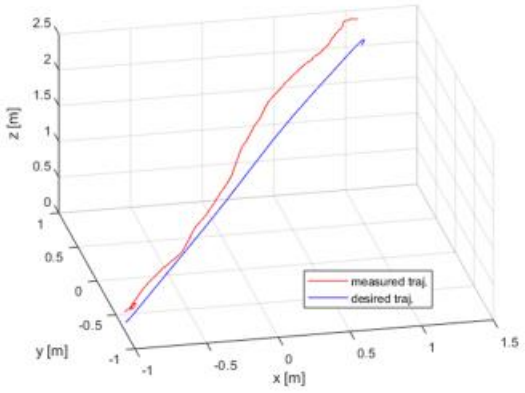
The drone has to move from $\boldsymbol{r}_0 = [-1\ \text{m} \quad -0.75\ \text{m} \quad 0.2\ \text{m}]^T$ to $\boldsymbol{r}_f = [1\ \text{m} \quad 0.75\ \text{m} \quad 2.2\ \text{m}]^T$ within $T = 3\ \text{s}$. In addition, it has to yaw so that it moves in its forward (x-) direction.

For path planning, the setpoints are changed at $t = T/2$.

Responses of experiment 1



Trajectories of experiment 1



Experiment 2: Simple Obstacle Avoidance

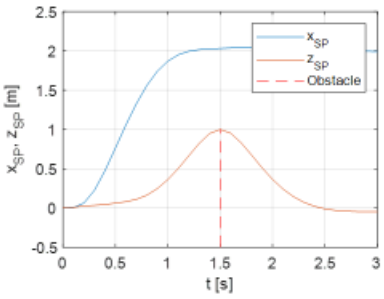
The drone has to move a distance of $2\ \mathrm{m}$ in x-direction within $T = 3\ \mathrm{s}$. It has to go over an obstacle (wall) of a height of $1\ \mathrm{m}$ „in the middle“ of its path.

The obstacle is modelled as state boundary: $r_z(T/2) \geq 1\mathrm{m}$

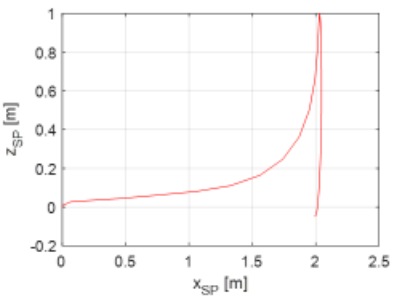
At first, the setpoint for r_x is changed at $t = 0$.

Experiment 2:

Time responses

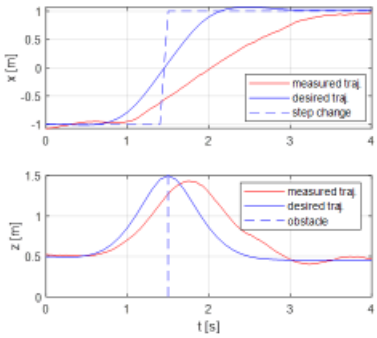


Trajectory

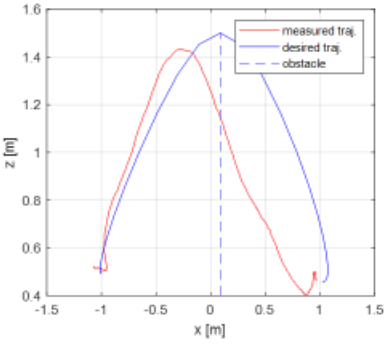


Experiment 2 with setpoint change for r_x $t = T/2$.

Time responses



Trajectory



Experiment 3: Improved Obstacle Avoidance

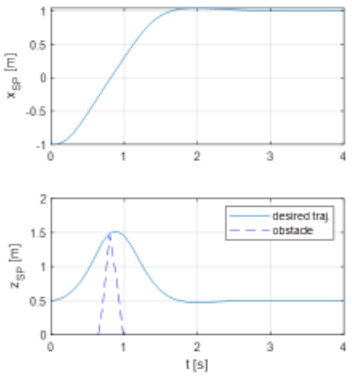
The drone has to move a distance of 2 m in x-direction within $T = 4$ s. It has to go over an obstacle (wall) of a height of 1 m in the middle of its path.

The obstacle is modelled as nonlinear state constraint $r_z(x) \geq z_{obst}(x)$, where $z_{obst}(x)$ is the difference of two sigmoid functions, slightly shifted relatively to one another with $\max(z_{obst}) = 1$ m.

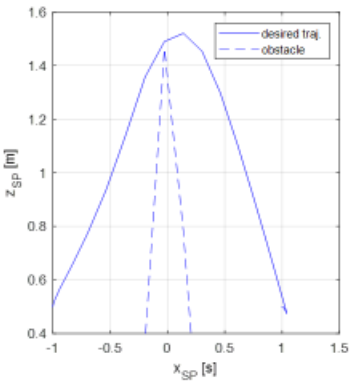
The setpoint for r_x is changed at $t = 0$.

Experiment 3:.

Time responses



Trajectory



Remark

In the path planning calculations, the constraints of the control variables are reached, in the experiments not.

Possible Improvements

- Retune onboard controller
- Consider onboard controller in path planning, i.e. use model of closed loop for MPC path planning

Both measures require good knowledge of the onboard controller.

Trial on Time Optimal Path Planning

Goal: The drone should travel along a predefined trajectory in minimal time.

The minimal time is restricted by the constraints of the forces (controls).

If the time is to be minimized, time is a decision variable in the optimization problem.

⇒ Time cannot be the independent variable any more.

It is reasonable to use the distance s travelled by the drone along the desired trajectory as independent variable in the description of the control problem.

⇒ A spatial description is required.

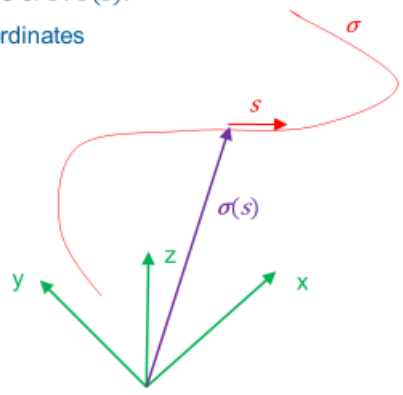
Desired Trajectory [1], [2]

The desired trajectory σ is a curve in 3-dim. space.

It is parametrized in dependance of s: $\sigma(s)$.

It can be represented by its coordinates

$$\boldsymbol{\sigma}(s) = \begin{bmatrix} \sigma_x(s) \\ \sigma_y(s) \\ \sigma_z(s) \end{bmatrix}$$



Spatial Differential Equation [1], [2]

The behavior of the system (drone) usually is described by the DE of order n

$$\dot{\boldsymbol{x}} = d\boldsymbol{x}/dt = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

where time t is the independent variable.

With $$\frac{dx}{dt} = \frac{dx}{ds} \cdot \frac{ds}{dt}$$

we get the spatial representation

$$\frac{d\boldsymbol{x}}{ds} = \frac{\boldsymbol{f}(\boldsymbol{x}(s), \boldsymbol{u}(s))}{ds/dt} = \frac{\boldsymbol{f}(\boldsymbol{x}(s). \boldsymbol{u}(s))}{\dot{s}}$$

where travelled distance s is the independent variable.

This spatial DE

$$\frac{d\boldsymbol{x}}{ds} = \frac{\boldsymbol{f}(\boldsymbol{x}(s), \boldsymbol{u}(s))}{ds/dt} = \frac{\boldsymbol{f}(\boldsymbol{x}(s). \boldsymbol{u}(s))}{\dot{s}}$$

must be extended by an equ. for time t:

$$\frac{dt}{ds} = \frac{1}{\dot{s}}$$

Obviously, the spatial description is of order n+1.

$\dot{s}$ is the projection of the drone's velocity onto the reference trajectory σ.

Calculation of $\dot{s}$ for Planar Motion (Motion of a Car) [1]

Known quantities: $\boldsymbol{\sigma}$, ψ_σ, ρ, $\boldsymbol{r}$, v_x, v_y

Vehicle velocity in direction of σ

$$v_\sigma = v_x \cos\psi_\sigma + \sin\psi_\sigma$$

Angular velocity about center of rotation

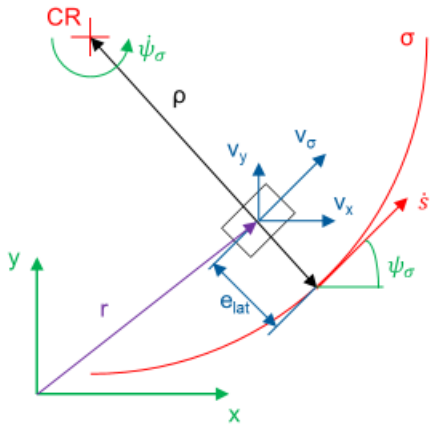
$$\dot{\psi}_\sigma = \frac{1}{\rho - e_{\text{lat}}} v_\sigma$$

Vehicle velocity projected on σ

$$\dot{s} = \rho \cdot \dot{\psi}_\sigma = \frac{\rho}{\rho - e_{\text{lat}}} v_\sigma$$

$$= \frac{1}{1 - e_{\text{lat}} * \kappa} v_\sigma$$

(Curvature $\kappa = 1/\rho$)

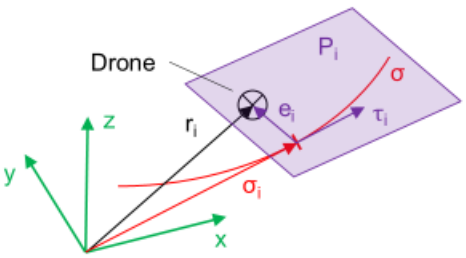


23 W. Krämer

Path Planning for the Drone
Time Optimal Path Planning

Calculation of $\dot{s}$ for Motion in 3-dimensional Space

Idea: Consider the spacial motion as a sequence of planar motions in local planes. Calculate $\dot{s}$ in the local planes.



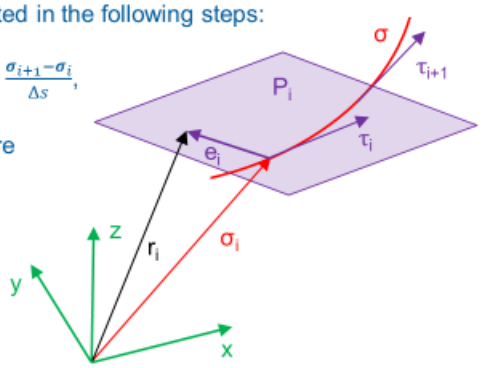
The local plane P_i at step i is defined by the desired position $\boldsymbol{\sigma}_i$, the tangent unit vector $\boldsymbol{\tau}_i$ on σ at $\boldsymbol{\sigma}_i$, and the position deviation $\boldsymbol{e}_i = \boldsymbol{\sigma}_i - \boldsymbol{r}_i$.

24 W. Krämer

Calculation of $\dot{s}$ for Motion in 3-dimensional Space (2)

At each instant i, $\dot{s}_i$ can be calculated in the following steps:

1. Calculate $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}(s_i)$, $\boldsymbol{\tau}_i = \frac{\partial \boldsymbol{\sigma}_i}{\partial s} \approx \frac{\boldsymbol{\sigma}_{i+1} - \boldsymbol{\sigma}_i}{\Delta s}$,
   and $\frac{\partial^2 \boldsymbol{\sigma}_i}{\partial s^2} \approx \frac{\boldsymbol{\tau}_{i+1} - \boldsymbol{\tau}_i}{\Delta s}$; $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_{i+1}$ are unit vectors.



2. Position deviation
   $\boldsymbol{e}_i = \boldsymbol{\sigma}_i - \boldsymbol{r}_i$.
   Now the plane P_i is defined:
   $P_i = \{p_i \in \mathbb{R}^3 \mid p_i = \sigma_i + \lambda e_i + \mu \tau_i, \quad \lambda, \mu \in \mathbb{R}\}$

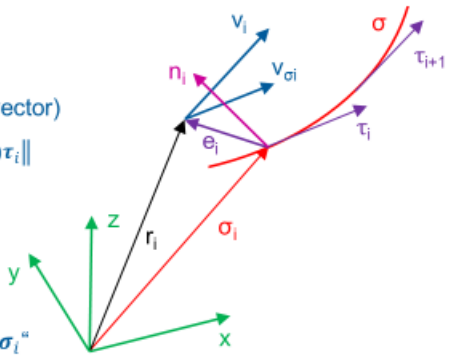
Path Planning for the Drone
Time Optimal Path Planning

Calculation of $\dot{s}$ for Motion in 3-dimensional Space (3)

3. Projection of velocity $\boldsymbol{v}_i$ on $\boldsymbol{\tau}_i$
   $v_{\sigma i} = \boldsymbol{v}_i^T \cdot \boldsymbol{\tau}_i$

4. Normal vector on $\boldsymbol{\tau}_i$ in P_i (unit vector)
   $\boldsymbol{n}_i = \boldsymbol{e}_i - (\boldsymbol{e}_i^T \cdot \boldsymbol{\tau}_i)\boldsymbol{\tau}_i / \|\boldsymbol{e}_i - (\boldsymbol{e}_i^T \cdot \boldsymbol{\tau}_i)\boldsymbol{\tau}_i\|$

5. Curvature
   $\kappa_i = \boldsymbol{n}_i^T \frac{\partial^2 \boldsymbol{\sigma}_i}{\partial s^2}$



$\kappa_i > 0$ if $\boldsymbol{r}_i$ is located "inside of $\boldsymbol{\sigma}_i$"
$\kappa_i < 0$ if $\boldsymbol{r}_i$ is located "outside of $\boldsymbol{\sigma}_i$"

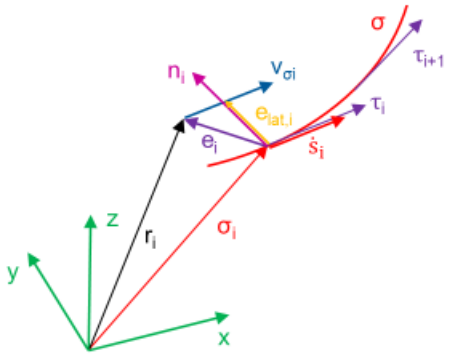
Calculation of $\dot{s}$ for Motion in 3-dimensional Space (4)

6. Lateral position deviation

$$e_{\mathrm{lat},i} = \boldsymbol{n}_i^T \cdot \boldsymbol{e}_i \geq 0$$

7. Projected velocity

$$\dot{s}_i = \frac{1}{1 - e_{\mathrm{lat},i} \cdot \kappa_i} \cdot v_\sigma$$



Optimization Problem (usual MPC control problem) [2]

$$\mathbb{P}_N(\boldsymbol{x}):\ V_N^0(x) = \min_{\boldsymbol{u}}\{V_N(\boldsymbol{x}(s), \boldsymbol{u}(s), t_N) | \boldsymbol{u} \in \mathcal{U}_N(x)\}$$

with

$$V_N(\boldsymbol{x}, \boldsymbol{u}) = \sum_{i=0}^{N-1} (\boldsymbol{x}_i - \boldsymbol{x}_{i,sp})^T \boldsymbol{Q}_i (\boldsymbol{x}_i - \boldsymbol{x}_{i,sp}) + (\boldsymbol{u}_i - \boldsymbol{u}_{i,sp})^T \boldsymbol{R}_i (\boldsymbol{u}_i - \boldsymbol{u}_{i,sp})$$
$$+ (\boldsymbol{x}_N - \boldsymbol{x}_{N,sp})^T \boldsymbol{Q}_f (\boldsymbol{x}_{N,sp} - \boldsymbol{x}_{N,sp}) + (t_N - T_{ref}) q_T (t_N - T_{ref})$$

and $\quad \boldsymbol{x}_{i,sp} = [\boldsymbol{\sigma}_{k,sp}^T \quad \ldots]^T, \qquad i = 0, \ldots, N$

$$d\boldsymbol{x}/ds|_i = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

$$\boldsymbol{u}_i \geq \boldsymbol{0}, \quad \boldsymbol{u}_i \leq \boldsymbol{u}_{\max}$$

Remark: For $T_{ref} < T^*$, the global optimum of the problem is $t_N = T^*$, where T^* is the mimimum time the drone requires to run along σ (Observation in [2]).

Bad News

The control problem based on the spatial description cannot be solved by mpctools.

Reasons:

- Division by zero, if $\dot{s} = 0$
- Difficulties in calculation of Jacobian due to taking norms.

mpctools seems not to be flexible enough to solve this problem.

Simplified Time Optimal Path Planning – Proposal of Jim

Apply time normalization to the system model $\dot{\boldsymbol{x}} = d\boldsymbol{x}/dt = \boldsymbol{f}(\boldsymbol{x}(t), u(t))$:

$\tau = t/t_f$ with final time t_f

New model for $\boldsymbol{z}(\tau(t)) = \boldsymbol{x}(t)$

$d\boldsymbol{z}/d\tau = \boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u}) * t_f$

Define $v = t_f$ as additional decision variable.

Then the optimization problem can be defined

$$\min_{\boldsymbol{u}, v} \left\{ \int_0^1 \boldsymbol{u}^T R \boldsymbol{u} \, d\tau + v \right\}$$

with terminal conditions

$\boldsymbol{z}(1) = \boldsymbol{z}_f$.

Path Planning for the Drone

Time Optimal Path Planning

The desired trajectory is defined by state constraints.

For this define a function

$$\gamma(\mathbf{z}_{ref}) = 0, \quad \gamma(\mathbf{z} \neq \mathbf{z}_{ref}) > 0$$

Then the state constraint would be

$$\gamma(\mathbf{z}) \leq \gamma_{\max}$$

with a tolerance $\gamma_{\max}$.

Question:

Is it possible to keep $v = t_f$ constant during a simulation run, when using mpctools?

Path Planning for the Drone

References

[1] Gao, Yiqi, Andrew Gray, Janick V. Frasch, Theresa Lin, Eric Tseng, J. Karl Hedrick, Francesco Borrelli: Spatial predictive control for agile semi-autonomous ground vehicles. Proceedings of the 11th international symposium on advanced vehicle control, no. 2, pp. 1-6. 2012.

[2] Verschueren, Robin, Stijn De Bruyne, Mario Zanon, Janick V. Frasch, Moritz Diehl: Towards time-optimal race car driving using nonlinear MPC in real-time. 53rd IEEE conference on decision and control, pp. 2505-2510. IEEE, 2014.

Thank you very much,

in particular to Jim ...,

to Koty ...,

and to all of you
for your attention!

And good bye!

Technische Hochschule Ingolstadt

Working Paper

Heft Nr. 56 aus der Reihe
„Arbeitsberichte – Working Papers"

ISSN 1612-6483
Ingolstadt, im Dezember 2020

Prof. Dr.-Ing. Wolfgang Krämer

Modellprädikative Regelung einer Drohne

Impressum

Herausgeber
Der Präsident der Technischen Hochschule Ingolstadt
Esplanade 10, 85049 Ingolstadt
Telefon: +49 841 9348-0
Fax: +49 841 9348-2000
E-Mail: info@thi.de

Druck
Hausdruck

Internet
Alle Themen aus der Reihe „Arbeitsberichte – Working Papers", können Sie unter der Adresse www.thi.de nachlesen.